

# Spiking Neural Networks in Intelligent Edge Computing

**Guanlei Zhang, Lei Feng, Fanqin Zhou, and Zhixiang Yang**  
Beijing University of Posts and Telecommunications

**Qiyang Zhang**  
Peking University

**Alaa Saleh**  
University of Oulu

**Praveen Kumar Donta**  
Stockholm University

**Chinmaya Kumar Dehury**  
University of Tartu

**Abstract**—Deep neural networks (DNNs) have witnessed rapid advancements and remarkable success in recent years, leading to their increasingly widespread implementation on edge devices. However, the deployment, execution, and life cycle management of traditional artificial neural networks (ANNs) on resource-constrained edge devices present significant challenges. Spiking neural networks (SNNs) are a class of neuroscience-inspired neural networks that emulate the low-power operational mode of biological neurons. SNNs possess advantages such as low power consumption, low latency, event-driven processing, and reduced communication overhead, making them particularly well-suited for edge devices and intelligent edge computing. As a result, they have garnered significant attention in both research and practical applications. In this paper, we present a comprehensive survey of the fundamentals of SNNs and the advancements in SNN research for edge computing, exploring potential applications and future directions in this emerging field. We also present a case study highlighting that SNNs outperform ANNs in distributed learning, achieving a 6% improvement in accuracy and an 80% reduction in data transmission.

■ **IN RECENT YEARS**, artificial intelligence (AI) has made rapid progress, with deep learning (DL), particularly deep neural networks (DNNs), recognized as a primary driving force. DNNs have become widely adopted due to their exceptional performance across different applications and domains, including image recognition, speech processing, and natural language processing [1]. As resource-constrained devices, such as consumer electronics, become increasingly prevalent, they also benefit from the advanced capabilities offered by DNNs. However, the training and inference of DNNs are computationally intensive tasks, pre-

senting significant challenges when running on these devices, which are typically constrained in terms of computation, storage, and energy resources [2].

Current research and applications primarily focus on artificial neural networks (ANNs), but their application in edge environments faces two main challenges. On one hand, ANNs depend on continuous real numbers and computationally expensive Multiply-Accumulate (MAC) operations to transmit information between layers. On the other hand, modern ANNs are frequently designed as deep models with multi-branch architectures, further increasing their complexity. As this complexity grows, so do the computational and energy demands [3].

*Digital Object Identifier 10.1109/MCE.2023.Doi Number*

*Date of publication 00 xxxx 0000*

In response, Spiking Neural Networks (SNNs), as an emerging generation of neuroscience-inspired neural network models, are attracting considerable interest [4]. SNNs use spatio-temporal neurodynamic models to simulate how neurons and synapses in the brain fire and interact. Compared to traditional ANNs, SNNs have key benefits like energy efficiency, asynchronous processing, and lower communication costs, which make them ideal for edge devices like wearables, smart home systems, and security applications. Therefore, it is crucial to explore the deployment of SNN-based intelligent applications on resource-constrained devices, as well as the integration of SNNs with intelligent edge computing [5]. We summarize the main contributions of this work as follows.

- We summarize the foundational concepts of SNNs and analyze their advantages and challenges in the context of edge computing.
- We review recent advancements in SNNs for edge intelligence and present a case study that provides detailed insights into the performance of SNNs in distributed learning scenarios.
- We offer direction for future research, focusing on overcoming current challenges and advancing the deployment of SNNs in edge environments.

This paper explores research gaps by analyzing the capabilities and applications of SNNs in intelligent edge computing scenarios. First, in the "OVERVIEW OF SNN" section, we provide an overview of the core concepts of SNNs, including the major challenges and benefits at the edge. Next, in the "SNNs IN THE EDGE: LITERATURE AND USE CASES" section, we review the research progress of SNNs in edge intelligence. We then present a case study using distributed learning with DNNs to demonstrate the promising future and practical value of SNNs in intelligent edge computing. In the "OPEN CHALLENGES AND FUTURE RESEARCH" section, we analyze the potential challenges and future research directions for SNNs in intelligent edge computing. Finally, in the "CONCLUSION" section, we summarize the key insights of the paper. Through this work, our objective is to provide a comprehensive reference for SNN-based edge intelligence research, driving the deployment and application of AI on resource-constrained devices.

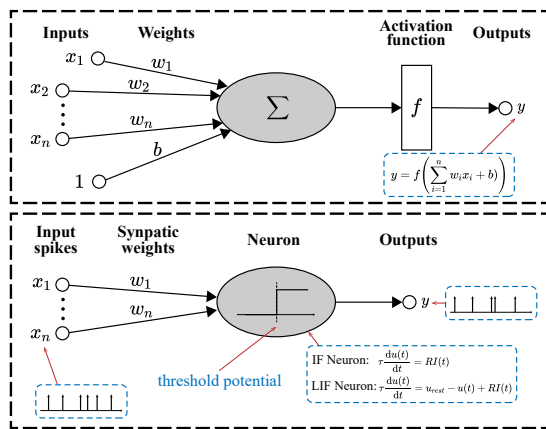
## OVERVIEW OF SNN

This section explains the basic concept of SNNs and their distinct biological background. SNNs are a

class of neural network models inspired by neuroscience, characterized by their ability to simulate the firing events and dynamics of biological neurons and synapses. The basic units of SNNs are neuron models, which have been proposed with varying degrees of biological fidelity. Some of commonly used models include the Integrate-and-Fire (IF), Leaky Integrate-and-Fire (LIF), and Spike Response Model (SRM). These models strike a good balance between biological fidelity and computational complexity under current optimization algorithms and are widely used in deep learning tasks such as image recognition. The more complex Hodgkin-Huxley model, derived from studies on the membrane potentials of squid giant axons. However, despite its higher biological realism, its computational complexity is too high for current optimization algorithms. Unlike traditional ANNs, information in SNNs is encoded and transmitted in the form of discrete spikes. Each input is presented within a predefined time window, meaning that SNNs typically require multiple forward propagations. Activation of the neuron sends a signal in the form of synaptic current to the next neuron, with the signal strength proportional to the weight. When the synaptic current reaches the target neuron, it changes the neuron's membrane potential. Most spiking neuron models share common characteristics: they have an internal state that accumulates input stimuli and generate a spike when this internal state exceeds a threshold. These elements give SNNs their unique advantages in simulating biological computation, enhancing energy efficiency, and enabling event-driven processing. In this work, we primarily discuss the popular IF/LIF neuron models, along with common encoding methods and learning strategies.

## Neuron Models and Encoding Methods

In traditional ANNs, neuron computation and output can be viewed as the combination of a linear function and a nonlinear activation function, as illustrated in the upper part of Figure 1. Information in ANNs propagates forward synchronously, layer by layer, in a stateless manner; each neuron performs a single nonlinear mapping from input to output within the spatial domain. In contrast, SNNs inherently operate in the spatio-temporal domain. Due to their use of neural dynamics models, the internal states of SNN neurons exhibit temporal correlations, allowing SNNs to be regarded as a generalized form of Recurrent Neural Networks. Additionally, information transfer



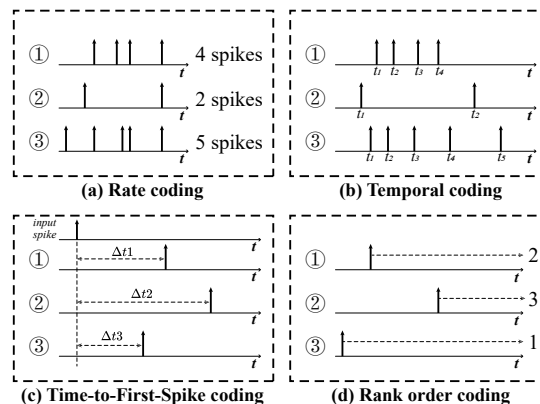
**Figure 1.** Comparison of ANN and SNN's Neuron models.

between neurons in SNNs is asynchronous—neurons do not need to wait for spikes from all neurons in the previous layer before transmitting information, enabling a depth-first propagation method.

**IF/LIF Neuron Models:** The widely used neuron models in recent years are the IF/LIF models, which strike a favorable balance between computational efficiency and biological plausibility. The structure of the IF/LIF neuron models is illustrated in the lower part of Figure 1.  $u(t)$  represents the membrane potential of the neuron, which is also the internal state of the neuron. Remaining parameters are as follows:  $\tau$  is the membrane time constant,  $u_{rest}$  is the resting potential of the neuron, and  $R$  and  $I(t)$  are the input resistance and input current of the cell membrane, respectively. The membrane potential accumulates input current over time. When this potential reaches a specific threshold, the neuron generates a spike and then resets to its baseline potential. In the LIF model, in the absence of input, the membrane potential gradually decays to the resting potential. The key difference between the IF model and the LIF model is that the IF model lacks potential decay, effectively functioning as a pure integrator. From a simulation perspective, the workflow of IF/LIF neurons can be understood as follows: the neuron integrates the weighted sum of incoming binary spikes, and once this sum exceeds its threshold, the neuron emits a spike to the next neuron. IF/LIF models greatly simplify the process of membrane potential variation, making their implementation and deployment within DL frameworks highly efficient.

**Encoding Methods:** ANNs represent and transmit information using real values in the spatial domain, whereas SNNs convey information in the temporal domain through binary spikes. This shift necessitates the development of appropriate coding strategies to encode spikes, specifically considering how to capture the temporal information contained in spikes arriving at different times. The two predominant neural coding methods are rate coding and temporal coding, as illustrated in Figure 2.

- **Rate Coding:** It utilizes multiple spikes to represent a single unit of information, primarily focusing on the mean firing rate within time period, as shown in Figure 2(a). This method does not consider the characteristics of individual spikes or the temporal structure between spikes, such as spike arrival times and intervals. Rate coding can be seen as a quantification of neuronal output, where the firing rate is analogous to the continuous output values in ANNs. Consequently, rate coding is widely used in the conversion of pre-trained ANNs to SNNs. A common rate coding method is Poisson encoding, where the spike firing rate is determined by the intensity of the pixel value: the higher the pixel value, the greater the corresponding spike firing rate.
- **Temporal Coding:** For efficient and energy-saving computation in practical neuromorphic hardware, using fewer spikes to represent information is advantageous, as it reduces energy consumption. Unlike rate coding which relies on spike firing rates, temporal coding focuses on the precise timing of



**Figure 2.** General structure of neural encoding methods in SSN.

individual spikes. This general concept is illustrated in Figure 2(b). Beyond this general form, specific temporal coding methods have emerged, each exploiting distinct temporal features of spikes. For instance, Time-to-First-Spike coding (Figure 2(c)), also known as latency coding, restricts each neuron to fire only once, with spike timing mapping to continuous values. In this approach, the delay of spikes is inversely proportional to the magnitude of the represented value. Another example is rank order coding, which considers the relative order of spikes rather than their precise timing (Figure 2(d)).

*Learning Strategies:* The learning process in neural networks is data-driven, involving the adjustment and determination of network weights to enable the network to perform specific tasks. Therefore, learning strategies are crucial. In general, SNNs employ two types of learning methods: 1) bio-inspired unsupervised learning; 2) supervised learning based on the backpropagation (BP) algorithm.

- **Bio-inspired Unsupervised Learning: Spike Timing Dependent Plasticity (STDP)** [6] is a popular unsupervised learning technique in SNNs. STDP is based on the neurobiological findings of Donald Hebb in 1949, which describe the learning process of biological neurons. STDP modifies synaptic weights based on the firing times of pre- and post-synaptic neurons. If a pre-synaptic neuron fires before a post-synaptic neuron, indicating an association between the neurons, the synaptic weight is increased. Conversely, if the pre-synaptic neuron fires after the post-synaptic neuron, indicating a weaker causal link, the synaptic weight is decreased.
- **Supervised Learning:** With the increasing availability of labeled data, supervised learning has become the most widely used learning strategy. In ANNs, the loss between the actual values and the network output is used to calculate gradients via the BP algorithm, which updates the network weights layer by layer. However, in SNNs, information is transmitted using discrete spikes, and the derivative of the spiking activation function is typically the Dirac delta function, leading to non-differentiability issues for traditional BP algorithms in SNNs. To address this, surrogate gradients have been proposed [7]. This approach involves using discrete spikes during the forward pass and replacing the spiking activation function with a differentiable approximation during

the backward pass, thereby enabling gradients to be backpropagated through the spatial-temporal dimensions of the network.

## SNNs for the Edge: Benefits and Challenges

*1) Advantages of SNNs for the Edge:* Compared to ANNs, SNNs demonstrate key advantages primarily in terms of computational and communication overhead, making them well-suited for the edge.

- **Low Cost and Latency:** As previously noted, SNNs operate without the need for complex mathematical multiplications and additions, instead functioning as integrators. The sparse binary spikes of SNNs significantly reduce computational demands and data transmission overhead. Additionally, the asynchronous nature of SNNs enables real-time responsiveness, leading to faster processing speeds and lower latency—features that are essential for time-sensitive applications. Given these advantages, SNNs are particularly well-suited for edge scenarios where minimizing latency is a priority.
- **Low-Communication Distributed Deployment:** In distributed neural networks, particularly in wireless settings with dense small cell deployments, minimizing the local overhead on resource-constrained devices is a growing concern. While distributed ANNs require the transmission of substantial amounts of real-valued intermediate data between nodes, which can overload network bandwidth and increase latency, SNNs address these issues by transmitting only sparse binary spikes. This drastically lowers energy consumption and communication latency, positioning SNNs as a more efficient choice for distributed deployment on edge devices.

*2) Challenges of intelligent edge computing:* We identify and summarize the two primary challenges faced in intelligent edge computing such as communication and computation environments.

- **Dynamic and Uncertain Wireless Environments:** Wireless communication scenarios are inherently dynamic [14], which renders the deployment of edge intelligence applications in such environments significantly more uncertain. These uncertainties stem from factors such as fluctuating network conditions, user mobility, and wireless communication interference. Traditional ANNs must frequently adjust deployment strategies and fine-tune model weights to mitigate the impact of wireless network

**Table 1. Summary of Literature on SNN in edge intelligence.**

Topics	References	Contributions
Topic 1 Edge Computing	[8]	Optimizing the energy-efficient deployment of SNNs across multiple edge nodes by jointly managing computation and communication resources
	[9]	Analyzing the impact of radio losses on performance in different spike communication schemes within SNNs
Topic 2 Distributed Learning	[10]	Federated Learning approach based on a probabilistic SNN model, balancing communication efficiency and accuracy
	[11]	Decentralized energy-efficient SNN approach with accuracy comparable to existing edge AI technologies
Topic 3 Neuromorphic Communication	[12]	End-to-end neuromorphic edge architecture integrating SNNs with edge intelligence and JSCC capabilities
	[13]	Neuromorphic communication system enabling joint training of encoding and decoding SNNs across multiple channels

fluctuations on model performance. Consequently, this necessity further exacerbates the substantial overhead associated with ANNs.

- **Constraints of Edge Devices:** Deploying neural network-driven intelligent applications on edge devices presents significant challenges due to various constraints such as energy, bandwidth, and computational capabilities. The efficient deployment of intelligent applications in edge scenarios remains a highly active area of ongoing research.
- **Trade-offs between Performance and Cost:** SNNs feature sparse and accumulative computations, which significantly reduce computational and communication costs. However, SNNs of comparable scale generally underperform ANNs in many tasks, necessitating careful trade-offs based on specific task requirements, particularly in edge-distributed scenarios. The choice between SNNs and ANNs thus depends largely on whether minimizing energy consumption and communication costs is more critical than achieving the highest accuracy.

## SNNS IN THE EDGE: LITERATURE AND USE CASES

In this section, we summarize the current research advancements of SNNs in the field of intelligent edge computing and analyzes their potential applications.

### Literature on SNN in Edge Computing

Currently, research on SNNs in the field of intelligent edge computing is still in its early stages, leaving many directions open for further investigation and exploration. Table 1 summarizes recent advancements and contributions of SNNs to intelligent edge computing, focusing on the integration of SNNs with distributed learning methods and the optimization of SNN deployment in distributed wireless scenarios.

1) *Optimized Deployment of SNN in Edge Com-*

*puting:* This line of research primarily focuses on optimizing the deployment of SNNs in wireless edge scenarios to enhance the quality of service for SNN-based intelligent applications. Liu et al. [8] and Borsos et al. [9] explored the optimization of distributed SNN deployment by partitioning SNNs and deploying them across network nodes. These problems that aim to minimize energy consumption and communication overhead by adjusting deployment locations, power levels, and bandwidth allocations.

2) *SNN with Distributed Learning:* Skatchkovsky et al. [10] and Yang et al. [11] integrated SNNs with the widely-used distributed learning framework federated learning (FL), utilizing different SNN models and FL methods. In [10], a Probabilistic SNN Model is utilized, with a focus on balancing communication overhead and accuracy through the selective exchange of weights. Yang et al. [11] eliminates the need for a central server by designating one device as the leader for model aggregation. This approach reduced data traffic and computational latency while maintaining near-optimal accuracy.

3) *Neuromorphic Communication:* Some studies have explored the application of SNNs in communication coding. Chen et al. [13] proposed an autoencoder architecture that integrates SNN-based edge intelligence with Joint Source-Channel Coding (JSCC). Similarly, [12] introduced the NeuroComm system, which employs SNNs to implement a form of JSCC and incorporates a hypernetwork to adjust SNN weights according to fading channel conditions.

### Use Cases

1) *Wearable Devices and Smart Healthcare:* Wearable devices and smart healthcare enable continuous monitoring and real-time data collection to enhance medical services [15]. Instead of transmitting large amounts of raw data, these systems focus on key health

metrics, reducing data volume. With low computational cost and energy-efficient spike-based processing, SNNs can further optimize wearable devices by minimizing data transmission, extending battery life, and enhancing real-time analytics, making them ideal for resource-constrained healthcare applications.

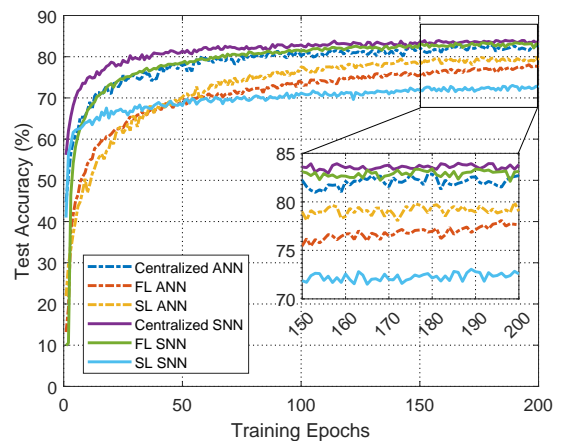
2) *Autonomous Vehicles*: SNNs hold significant potential in the field of autonomous vehicles. Their low energy consumption and real-time processing capabilities make them well-suited for resource-constrained edge devices, such as those used for sensor data processing and decision-making in autonomous vehicles. SNNs can effectively handle high-frequency signal processing and rapid response requirements in dynamic environments, including tasks like object detection and lane-keeping.

3) *Personalized AI Services*: SNNs will be considered a good choice for edge-based personalized AI agents due to their real-time processing and ability to handle temporal information and sequence tasks. Furthermore, their event-driven nature reduces computational overhead, making them energy-efficient. These capabilities enable SNN-based edge personalized AI agents to better understand their users' environment and context, thereby delivering highly intelligent, more efficient, and personalized user experiences and conserving energy, making SNNs ideal for applications that require immediate responses such as smart home agents. For example in smart homes, SNN-powered agents can enhance natural language understanding, enabling more intuitive voice interactions, immediate control over devices, and proactive assistance by learning user preferences.

4) *Data Privacy*: SNNs hold significant potential in the domain of data privacy. Their low energy consumption and real-time processing capabilities make them well-suited for resource-constrained edge devices used in sensor data processing and decision-making within autonomous vehicles. SNNs are adept at handling high-frequency signal processing and rapid response requirements in dynamic environments, such as tasks involving object detection and lane-keeping.

## CASE STUDY: SNN IN DISTRIBUTED LEARNING

In this section, we explore a specific use case of SNNs within the realm of intelligent edge computing. This use case demonstrates the application of SNNs in two classical distributed learning paradigms: split



**Figure 3.** Test accuracy of SNNs and ANNs under different training methods in image classification.

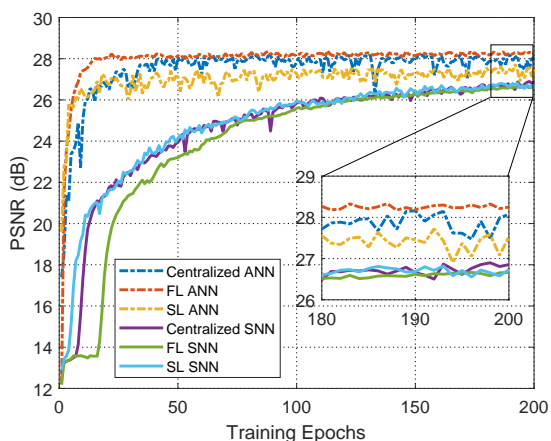
learning (SL) and FL.

### Implementation of SNN in Distributed Learning

To demonstrate the practical applications of SNNs in distributed edge computing, we evaluate their performance on two distinct AI tasks: image classification (high-level) and  $2\times$  super-resolution (low-level).

**Experimental Preparation:** 1) *Datasets*: For the image classification task, we utilize the CIFAR-10 dataset, which comprises images of ten classes of real-world objects. For the super-resolution task, we train using the DIV2K dataset, which contains 800 2K-resolution images of various scenes, and evaluate using the BSD100 dataset. 2) *Neural network models*: To ensure a fair comparison, we maintain consistent model for ANNs and SNNs. We designed a shallow 4-layer neural network to simulate its performance in edge environments. For the image classification task, the first two layers are convolutional layers with 128 channels, followed by two fully connected layers. The first fully connected layer compresses the feature representation by half, while the final layer outputs the classification results. For the super-resolution task, the network consists of four convolutional layers, with the first three layers having channels of 128, 128, and 256, respectively. The upsampling is achieved using pixel shuffle for  $2\times$  scaling, and the final layer reconstructs the RGB image. The SNN employs LIF neuron model, utilizing the arctangent function as a surrogate gradient during backpropagation. In contrast, the ANN uses the ReLU activation function.

**Training Process:** We implemented the simulation

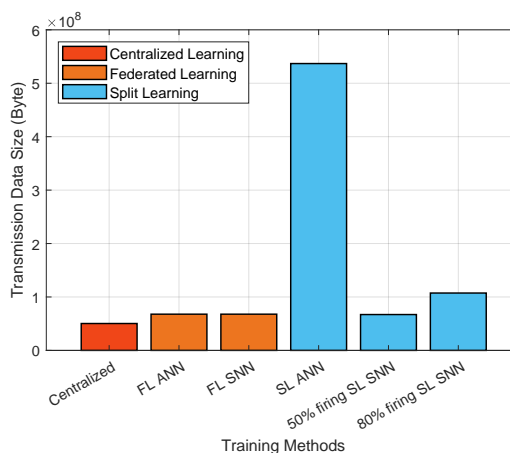


**Figure 4.** Test PSNR of SNNs and ANNs under different training methods in  $2\times$  super-resolution.

using PyTorch. For the image classification task, SNNs use mean squared error (MSE) loss function with one-hot encoding, while ANNs use cross-entropy loss function. For the super-resolution task, SNNs use MSE loss function, whereas ANNs employ L1 loss function. In the distributed learning setting, the number of users is set to 5 for the classification task and 3 for the super-resolution task, with local training epochs of 1 and 5, respectively, for a total of 200 training epochs. Accuracy is used as the metric for the classification task, while peak signal-to-noise ratio (PSNR) is used as the metric for the super-resolution task. The SNN is updated using backpropagation through time with a time window of 8, and both networks are optimized using the Adam optimizer with a learning rate of 0.001 and a batch size of 32.

### Numerical Analysis

Figure 3 presents the performance of SNNs and ANNs on the image classification task using different learning methods. Firstly, under traditional centralized learning, the accuracy of SNNs and ANNs is nearly identical, with SNNs outperforming ANNs by approximately 1%. Although ANNs are generally considered superior to SNNs in performance, this result indicates that when executing high-level tasks with the same shallow network, ANNs do not hold a clear advantage over SNNs, highlighting the potential benefits of deploying SNNs on edge devices. The results reveal that FL has minimal impact on SNN performance in image classification. However, SL causes a significant 11% accuracy drop compared to centralized learning.



**Figure 5.** Data transmission size under different training methods for  $32 \times 32$  RGB input images with a batch size of 4096.

Therefore, for this task, FL is a preferable learning method for edge scenarios. The results shown in Figure 4 contrast with those in Figure 3. In the low-level task of super-resolution, although ANNs exhibit a slight advantage due to the use of continuous real-valued data, achieving 1-2 dB higher PSNR than SNNs across different learning methods at the pixel level, SNNs demonstrate more stable performance in distributed training. Unlike ANNs, SNNs do not experience performance degradation or fluctuations caused by distributed learning, highlighting their advantages in edge-based distributed computing. Figure 5 illustrates the data transmission requirements for SNNs and ANNs under different learning methods, reflecting the communication load given specific channel conditions. Since the model parameters for both SNNs and ANNs are identical, the amount of data transmitted in FL is the same for both. In contrast, SL involves transmitting intermediate data. Under this learning method, SNNs benefit from data sparsity. Compared to the real-valued data transmitted by ANNs, SNNs transmit approximately one-fifth of the data, even at an 80% firing rate. Based on the results above, a clear trade-off between the performance and cost of SNNs and ANNs can be observed. In the case of SL, SNNs demonstrate significant advantages in reducing communication overhead; however, they may experience substantial performance loss in high-level tasks. In FL, the performance of SNNs remains largely stable, with only a slight gap compared to ANNs, achieving a favorable balance between performance and computational cost.

## OPEN CHALLENGES AND FUTURE RESEARCH

### Neuromorphic Datasets and Programming Tools

The development of neuromorphic datasets and programming tools is still limited. Existing datasets are primarily static, lacking the dynamic temporal information required by SNNs, which hinders their performance relative to ANNs. Additionally, unlike ANNs, which benefit from mature platforms such as PyTorch and TensorFlow, SNNs lack dedicated frameworks tailored to their unique architecture. As a result, researchers are often forced to adapt ANN-based frameworks for SNN simulations, adding complexity to the development process and limiting the seamless deployment of SNNs in practical scenarios.

### Efficient SNN Training Methods

While bio-inspired unsupervised training algorithms like STDP offer advantages in biological interpretability and training cost, they are not well-suited for large-scale problems such as ImageNet, and their accuracy lags behind gradient-based methods. In contrast, supervised training approaches benefit from the introduction of supervisory signals, which enhance model performance and structural scalability. However, due to the inherent spatio-temporal characteristics of SNNs, BP algorithms often suffer from extended training times and vanishing gradients, limiting their effectiveness in large-scale deep networks. As a result, developing more efficient SNN training techniques remains a significant challenge, particularly methods that integrate the strengths of both STDP and BP.

### Advanced SNN Model Design

Designing advanced SNN architectures remains a challenge, as the diverse service demands in edge environments require a careful balance between accuracy and efficiency. Future research can draw on mature architectural designs from ANNs, adapting and integrating them into SNNs for improved performance. Additionally, ANN-SNN hybrid architectures offer a promising research direction for balancing performance and computational cost, but they also introduce greater complexity in training methods.

### Neuromorphic Hardware

Neuromorphic hardware offers extensive parallelism and ultra-low-power solutions for the event-driven computation of SNNs. While some prototype

developments have emerged in recent years, such as IBM's TrueNorth and Intel's Loihi neuromorphic chips, these efforts are still in the early stages, and neuromorphic hardware remains significantly behind the development of ANN accelerators like GPUs and NPUs. Continuous exploration is required to innovate neuromorphic architectures, enhance energy efficiency, and reduce latency.

## CONCLUSION

As a new generation of neural networks, SNNs hold significant potential for advancing AI applications on edge and resource-constrained devices, due to their low computational and communication overhead. In this work, we explore recent advancements in SNNs, covering neuron models, encoding techniques, and learning strategies. We also review key research in the domain of edge intelligence and present a case study that provides insights into the performance of SNNs in distributed learning environments. Lastly, we examine the advantages and the future challenges of SNNs in edge intelligence. Through this study, we aim to provide a foundation for further research into the application of SNNs on resource-constrained devices.

## REFERENCES

1. J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proc. IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019.
2. S. Duan, D. Wang, J. Ren, F. Lyu, Y. Zhang, H. Wu, and X. Shen, "Distributed artificial intelligence empowered by end-edge-cloud computing: A survey," *IEEE Commun. Surv. Tut.*, vol. 25, no. 1, pp. 591–624, 2022.
3. Q. Zhang, X. Che, Y. Chen, X. Ma, M. Xu, S. Dustdar, X. Liu, and S. Wang, "A comprehensive deep learning library benchmark and optimal library selection," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 5069–5082, 2024.
4. M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: Opportunities and challenges," *Front. Neurosci.*, vol. 12, p. 409662, 2018.
5. Y. Venkatesha, Y. Kim, L. Tassioulas, and P. Panda, "Federated learning with spiking neural networks," *IEEE Trans. Signal Process.*, vol. 69, pp. 6183–6194, 2021.
6. S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, "Std-based spiking deep convolutional neural networks for object recognition," *Neural Netw.*, vol. 99, pp. 56–67, 2018.

7. E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Process. Mag.*, vol. 36, no. 6, pp. 51–63, 2019.
8. Y. Liu, Z. Qin, and G. Y. Li, "Energy-efficient distributed spiking neural network for wireless edge intelligence," *IEEE Trans. Wireless Commun.*, vol. 23, no. 9, pp. 10 683–10 697, 2024.
9. T. Borsos, M. Condoluci, M. Daoutis, P. Haga, and A. Veres, "Resilience analysis of distributed wireless spiking neural networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2022, pp. 2375–2380.
10. N. Skatchkovsky, H. Jang, and O. Simeone, "Federated neuromorphic learning of spiking neural networks for low-power edge intelligence," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2020, pp. 8524–8528.
11. H. Yang, K.-Y. Lam, L. Xiao, Z. Xiong, H. Hu, D. Niyato, and H. V. Poor, "Lead federated neuromorphic learning for wireless edge artificial intelligence," *Nature Commun.*, vol. 13, no. 1, p. 4269, 2022.
12. N. Skatchkovsky, H. Jang, and O. Simeone, "End-to-end learning of neuromorphic wireless systems for low-power edge artificial intelligence," in *Proc. Asilomar Conf. Signals, Syst., Comput.*, 2020, pp. 166–170.
13. J. Chen, N. Skatchkovsky, and O. Simeone, "Neuromorphic wireless cognition: Event-driven semantic communications for remote inference," *IEEE Trans. Cogn. Commun. Netw.*, vol. 9, no. 2, pp. 252–265, 2023.
14. R. V. W. Putra and M. Shafique, "Spikedyn: A framework for energy-efficient spiking neural networks with continual and unsupervised learning capabilities in dynamic environments," in *Proc. 58th ACM/IEEE Design Autom. Conf.*, 2021, pp. 1057–1062.
15. S. Shajari, K. Kuruvashetti, A. Komeili, and U. Sundararaj, "The emergence of ai-based wearable sensors for digital health technology: A review," *Sensors*, vol. 23, no. 23, p. 9498, 2023.

**Guanlei Zhang** is currently with the Beijing University of Posts and Telecommunications, Beijing, China. Contact him at zhangguanlei@bupt.edu.cn.

**Qiyang Zhang** is currently an postdoctoral researcher with the School of Computer Science, Peking University. Contact him at

qiyangzhang@pku.edu.cn.

**Lei Feng** is currently an associate professor with the School of Computer Science, Beijing University of Posts and Telecommunications (BUPT). He is the corresponding author of this paper. Contact him at fenglei@bupt.edu.cn.

**Alaa Saleh (S'23)** is currently doctoral researcher at Center for Ubiquitous Computing, University of Oulu, Finland. Contact her at alaa.saleh@oulu.fi.

**Dr. Praveen Kumar Donta (SM'22)** is currently senior lecturer at Department of Computer and Systems Sciences, Stockholm University, Sweden. Contact him at praveen@dsv.su.se.

**Dr. Chinmaya Kumar Dehury (M'17)** is currently assistant professor at Institute of Computer Science, University of Tartu, Estonia. Contact him at chinmaya.dehury@ut.ee.

**Fanqin Zhou** is currently a lecturer with the School of Computer Science, Beijing University of Posts and Telecommunications (BUPT). Contact him at fqzhou2012@bupt.edu.cn.

**Zhixiang Yang** is currently a a post-doctoral researcher with the School of Computer Science, Beijing University of Posts and Telecommunications (BUPT). Contact him at yangzx@bupt.edu.cn.